Elmar Schömer
Ann-Christin Wörl

4. Übungsblatt

Abgabe: Dienstag, der 21.11.2023, 14:00 Uhr

Aufgabe 1: Sortierung (10 + 10 Punkte)

Gegeben sei ein Feld $a = [a_0, a_1, a_2, \ldots, a_{n-2}, a_{n-1}]$.
Schreiben Sie Python-Programme, die die Elemente des Feldes a (in-place) umordnen, d.h. ohne Zuhilfenahme eines weiteren Feldes.

1. Es soll die umgekehrte Reihenfolge der Elemente entstehen: $a = [a_{n-1}, a_{n-2}, \ldots, a_2, a_1, a_0]$
2. Die Elemente sollen zyklisch um eine Feldposition nach links geshiftet werden:
   $a = [a_1, a_2, \ldots, a_{n-2}, a_{n-1}, a_0]$

Aufgabe 2: Skat (5 + 5 + 10 + 10* Punkte)

Das folgende Programm erzeugt eine zufällige Permutation der Symbole ♣, ♠, ♡, ♢ so dass alle $4! = 24$ Permutationen gleich wahrscheinlich sind (siehe https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle#The_modern_algorithm).

```
1  import random
2  p = ['Kreuz','Pik','Herz','Karo']
3  for i in range(3,0,-1):
4      j = random.randint(0,i)
5      tmp = p[i]
6      p[i] = p[j]
7      p[j] = tmp
8  print(p[0]+' '+p[1]+' '+p[2]+' '+p[3])
```

1. Beschreiben Sie in Worten, wie es dem Algorithmus gelingt, eine Permutation der Elemente des Feldes p herzustellen. Begründen Sie, dass jede der 24 Permutationen mit der gleichen Wahrscheinlichkeit auftritt.
2. Weisen Sie experimentell nach, dass die Permutation (♢, ♠, ♣, ♡) (genau wie jede andere) mit Wahrscheinlichkeit $\frac{1}{24}$ auftritt, indem Sie 1000000 zufällige Permutationen erzeugen und die Häufigkeit von (♢, ♠, ♣, ♡) ermitteln.

Das folgende Programm erzeugt die Symbole der Karten eines Skatblattes (siehe https://de.wikipedia.org/wiki/Skat), sortiert nach den Farben in der Reihenfolge Kreuz, Pik, Herz und Karo und innerhalb der Farben in der Reihenfolge Bube, As, König, Dame, 10, 9, 8, 7.

```
1  farbe = [0x1f0d0,0x1f0a0,0x1f0b0,0x1f0c0]
2  rang = [11,1,14,13,10,9,8,7]
3  for f in farbe:
4      for r in rang:
5          print(chr(f+r),end='')
6      print()
7  print(chr(0x1f0a0))
```

3. Schreiben Sie ein Programm, das ein Python-Feld mit allen 32 Spielkarten aufbaut und dieses mischt. Anschließend sollen die Karten auf drei Spieler-Felder der Länge 10 und das Skat-Feld der Länge 2 aufgeteilt und dargestellt werden.
4. **Zusatzaufgabe:** Ordnen Sie die Karten jedes Spielers in der oben beschriebenen Reihenfolge.

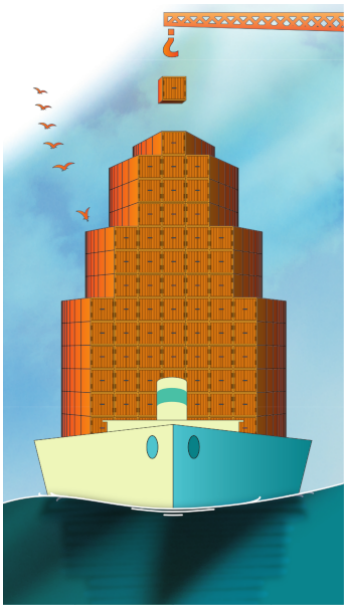
Aufgabe 3: Lotto (20 Punkte)

Wir wollen die Ziehung der Lottozahlen 6 aus 49 (ohne Superzahl) (siehe https://de.wikipedia.org/wiki/Lotto#Lottosysteme) simulieren.

1. Tippen Sie 6 Zahlen.
2. Schreiben Sie ein Python-Programm, das 6 Zahlen aus den Zahlen von 1-49 ohne Zurücklegen zieht. Achten Sie darauf, dass alle Ziehungen gleich wahrscheinlich sind.
3. Geben Sie die 6 gezogenen Zahlen in aufsteigender Reihenfolge aus.
4. Ermitteln Sie, wieviele richtige Zahlen Sie selbst getippt haben.

Aufgabe 4: Container (BWINF) (20 Punkte)

Junioraufgabe 2 des 41. Bundeswettbewerbs Informatik: Container
Mehrere Container werden per Kran auf ein Schiff verladen. Der schwerste Container soll zuerst auf das Schiff platziert werden; das ist bei Seegang gut für den Schwerpunkt.
Jeder Container ist vorher mindestens einmal mit einem anderen Container auf der Container-Paar-Waage gewesen. Für jedes gewogene Container-Paar ist danach nur bekannt, welcher der beiden Container der schwerere ist. Von den Containern haben keine zwei das gleiche Gewicht.

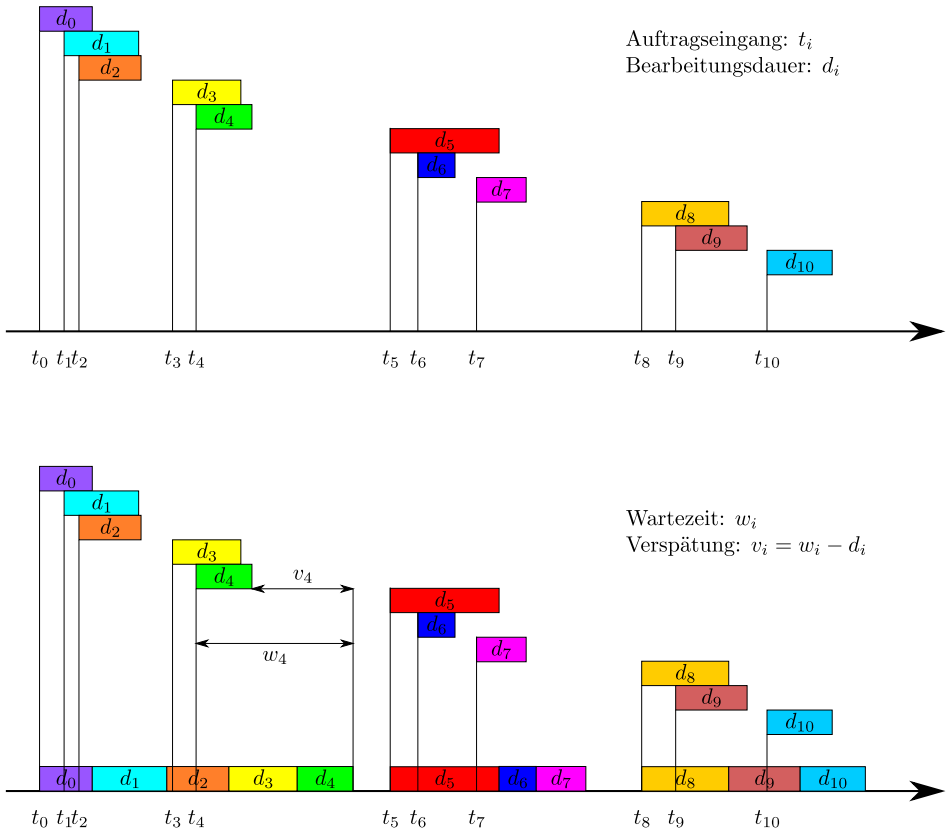


BUNDES WEIT INFORMATIK NACHWUCHS FÖRDERN
Bundeswettbewerb Informatik

Schreibe ein Programm, das die Kranführerin unterstützt. Das Programm soll für die gewogenen Container-Paare einlesen, welcher Container jeweils der schwerere ist. Anschließend soll das Programm ausgeben, ob daraus eindeutig der schwerste Container bestimmt werden kann. Falls ja, soll das Programm außerdem ausgeben, welcher Container das ist. Zu dieser Aufgabe gibt es 5 Beispieleingaben: `container?.txt`. In einer Zeile stehen jeweils die Nummern der zusammen gewogenen Container. Dabei ist der zuerst aufgeführte Container der schwerere Container des Containerpaares.
Wende dein Programm auf all diese Beispiele an.

Aufgabe 5: Fahrradwerkstatt (BWINF) (20 + 10* Punkte)

In Anlehnung an die Aufgabe 4 "Fahrradwerkstatt" des 41. Bundeswettbewerbs Informatik.
"Marc hat sich mit einer kleinen Fahrradwerkstatt selbstständig gemacht. Aufgrund des Fahrrad-Booms wird es zunehmend schwierig, die ganzen Reparaturaufträge von Kunden abzuarbeiten. Daher arbeitet Marc bereits jeden Tag von 9 bis 17 Uhr. Wenn er einen Auftrag beginnt, erledigt er ihn vollständig, bevor er einen neuen Auftrag beginnt. Am Ende eines Arbeitstages unterbricht er gegebenenfalls den aktuellen Auftrag und nimmt ihn am nächsten Arbeitstag wieder auf. Er erledigt die Aufträge einfach in der Reihenfolge des Eingangs. ..."
Die folgende Grafik verdeutlicht an einem Beispiel seine Vorgehensweise. Mit t_i bezeichnen wir den Zeitpunkt, zu dem der i-te Auftrag eingeht. d_i steht für die Bearbeitungsdauer. Die Bearbeitung des i-te Auftrages beginnt sobald alle vorherigen Aufträge abgearbeitet wurden. Mit w_i bezeichen wir die Wartezeit des i-ten Auftrages. Sie entspricht der Differenz zwischen dem Endzeitpunkt der Bearbeitung und dem Eingangszeitpunkt.



Die Beispieldateien (`fahrradwerkstatt?.txt`) geben zeilenweise den Eingang t_i (in Minuten ab t_0) und die Dauer d_i (in Minuten) an.

- Schreiben Sie ein Python-Programm, das die Werte t_i d_i zeilenweise von den Beispieldateien (`fahrradwerkstatt?.txt`) einliest und jeweils die maximale Wartezeit $w_{max} = \max\{w_i \,|\, 0 \leq i < n\}$ und die durchschnittliche Wartezeit $w_{mean} = \frac{1}{n}\sum_{i=0}^{n-1} w_i$ berechnet. Gehen Sie der Einfachheit halber davon aus, dass Marc durchgängig arbeitet.
- Passen Sie Ihr Python-Programm so an, dass Marc seine vorgegebene Arbeitszeit einhält. Er arbeitet jeden Tag von 09 bis 17 Uhr, hat also kein Wochenende und beachtet keine Feiertage. Der Startzeitpunkt t_0 entspricht dabei dem Start eines Kalendertages um 0 Uhr.